Applying Formal modelling to detect DoS attacks in wireless medium

Kashif Saghar* William Henderson* David Kendall* Ahmed Bouridane*†
* School of Computing, Engineering and Information Sciences, Northumbria University,
Pandon Building, Newcastle upon Tyne NE2 1XE, United Kingdom
† College of Computer and Information Sciences, King Saud University
P.O.Box 51178 Riyadh 11543, Kingdom of Saudi Arabia
kashif.saghar@unn.ac.uk

Abstract— **Due to broadcast transmission and unattended nature, and hostile environments a variety of denial of service (DoS) attacks are possible in both Wireless Sensor Networks (WSNs) and ad-hoc networks. We have developed a formal framework which can automatically verify different wireless routing protocols against DoS attacks exhaustively. In this paper we apply our formal framework against a secure ad-hoc routing protocol ARAN, which employs public cryptographic signatures as a defense against attacks. Our framework confirmed that ARAN is still vulnerable to different DoS attacks such as black hole, INA and wormhole. The framework also traces back the reason(s) as to why and how the attacks were successful.**

I. Introduction

Wireless Sensor Network (WSN) and ad-hoc networks are able to operate inattentively under any environmental conditions. The aim of these wireless systems is to gather data from remote locations and route the data wirelessly to secure places (i.e. base station). The nodes that gather data are called '*sources*' whereas the interested nodes are called '*sinks*'. The process of moving data from a source to a sink is called '*routing*'. The broadcast nature of radio transmission; limited computing, power and communication resources; unattended and potentially hostile nature of the environment are all major challenges to be overcome in this process of data routing.

The intruders in the hostile environments can launch different Denial of Service (DoS) attacks to prevent data from reaching the base station. As in the case of our recent publications [8], [9], we use the term *Denial of Service* in a general sense to mean the adverse effect of any malicious external agent (attacker) on the correct or timely delivery of data from the source nodes to the sink nodes. Our particular focus is on the effects of denial of service on routing protocols in WSNs. A brief summary of attacks that are considered has already been presented in our recent work [8] and more detailed survey of attacks are available in [6].

Formal modelling is a potent technique that can verify a system against every single execution trace and thus can confirm that the whole system functions as expected. Our research work is based on applying formal modelling to different routing protocols to detect their vulnerability against different DoS attacks. By developing formal models of routing protocols and attack's models and applying different LTL properties, attacks in many routing protocols can automatically be detected. The modelling tools we employ, UPPAAL [2] and SPIN [5], can generate the trace confirming how and why a particular attack was successful in a given protocol. In this paper we consider a well known ad-hoc secure routing protocol ARAN [10] and apply our formal modelling techniques to detect its vulnerability against DoS attacks. Since applying public cryptography in WSNs is becoming widespread, ARAN can also be employed as a routing protocol. It is worth noting that we are not interested in problems dealing with cryptographic aspects of ARAN, rather we are looking at the routing aspects and DoS attacks at the network layer. Our formal model confirms that ARAN indeed is vulnerable to attacks such as Invisible node attack (INA), black hole, and wormhole despite the use of expensive public key cryptography to prevent these attacks.

The rest of this paper is organized as follows: Section II briefly discusses related work; Section III breifly describes our methodology in general while ARAN protocol is described in Section IV. The formal framework developed and the results obtained are described in Section V; conclusions and further work are given in Section VII.

II. Related Work

It has been realized by researchers that computer simulation is often inadequate for finding errors in routing protocols and the development of formal models to check various aspects of routing protocols is increasingly becoming an important tool. Formal models have also been used in the analysis of security protocols. Different hidden attacks have been discovered using formal modelling e.g. [12] discovered faults in TinySec and LEAP protocols; the formal analysis in [11] disclosed successful attacks in SNEP, which is the basic component of the security protocol 'Security Protocols for Sensor Networks' (SPINS); the work on [3] developed a new approach Sledge and verified μTESLA and LEAP protocols using this method. The most noteworthy work is presented in [1] where the authors used SPIN [4] to analyse the effect of some DoS attacks on ad-hoc routing protocols based on Dynamic Source Routing (Ardiadne and endairA). The authors in [1] have used an automated security evaluation process and analysed all topologies for networks of up to 5 nodes. This work was extended in [8], [9] to check WSN protocols rigorously against DoS attacks and a modeling

978-1-86135-369-6/10/$25.00 ©2010 IEEE

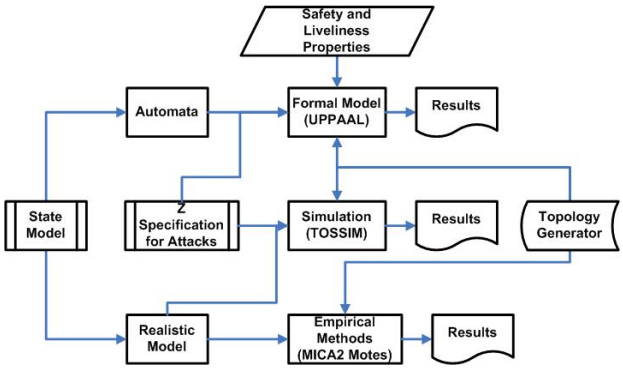

Fig. 1. Method adopted

framework was developed. This framework allows for an automatic analysis and vulnerability of some routing protocols to different DoS attacks using formal modeling and theorems (LTL properties).

III. Method adopted

As stated earlier a formal framework was developed by the authors [8] to verify the vulnerability of a variety of routing protocols against several Denial of Service (DoS) attacks. The method adopted to check different routing protocols is shown in figure 1. We generated high level and low level models from the state model of a routing protocol. The high level model was converted into a formal model and LTL properties (or theorems) exhaustively check the absence of any faults (attacks) present in the routing protocol. The properties included liveliness (something good will eventually occur), safety (nothing bad ever occurs) and basic sanity checks (confirmation that the model behaves as desired). In case a property fails, the formal model automatically generates a trace giving the user the reason as to how and why the attack occurred in the protocol. The low level model is then used to implement the protocol in a low level simulator (and perhaps a practical implementation on hardware). We used the same topology to confirm that the results of our formal model match the simulation results.

IV. ARAN Routing Protocol

ARAN [10] uses public key cryptography to ensure the integrity of routing messages. It is based on finding the quickest paths instead of shortest ones. This means that ARAN avoids hop counts to discover the routes. Initially, a source node S begins a route discovery process by broadcasting a route discovery message in the name of target node T:

$$S \rightarrow * : (RDP, T, cert_S, N_S, t, Sig_S)$$

Here RDP means that this is a route discover phase, S and T are the identifiers of the source and the target, respectively, N_S is a nonce generated by node S, t is the current time-stamp, $cert_S$ is the public-key certificate of the source, and Sig_S is the signature of the source on all of these elements. Later, as the request is propagated in the network, intermediate nodes also attach their signatures. Therefore, upon receiving an RDPS, a node A transmits the following message:

$$A \rightarrow * : (RDP, T, cert_S, N_S, t, Sig_S, Sig_A, cert_A)$$

When a neighbor of A (e.g. B) receives this route request, it verifies the signatures and the freshness of the nonce. If the verification is successful, then B updates its routing table for the source node S with A being the next hop node. Node B then replaces the certificate and the signature of A with its own and rebroadcast:

$$B \rightarrow * : (RDP, T, cert_S, N_S, t, Sig_S, Sig_B, cert_B)$$

A target node T, upon receiving the first route request verifications, updates its routing table as done by the previous nodes. T then unicast a route reply message (RREP) to source node S in the reverse path (node B) of the discovered route:

$$T \rightarrow B : (RREP, S, cert_T, N_S, t, Sig_T)$$

Here N_S and t are the nonce and the time-stamp obtained from the RDP message, respectively. S, $cert_T$ and Sig_T are the identifier of the source, the public-key certificate of T and the signature of T on all of these elements, respecyively. Similar to RDP, RREP is also signed by the intermediate nodes, too. Hence, the route reply sent by B to A is:

$$B \rightarrow A : (REP, S, cert_T, N_S, t, Sig_T, Sig_B, cert_B)$$

Upon receiving the RREP, node A verifies both signatures. If both are valid, A forwards the RREP in reverse path after replacing the certificate and the signature of B, with its own, in the message:

$$A \rightarrow S : (REP, S, cert_T, N_S, t, Sig_T, Sig_A, cert_A)$$

Node A also updates its routing table for the target node T with B being the next hop. Although, nothing has been said about data propagation we believe it has the same pattern as those of RREP. For example, suppose source node is sending data to target T via the verified path node A:

$$S \rightarrow A : (DATA, T, cert_S, N_S, t, Sig_S)$$

The nodes then again follow the data in the same pattern:

$$A \rightarrow B : (DATA, T, cert_S, N_S, t, Sig_S, Sig_A, cert_A)$$

If a link is broken or no traffic flows in an active route a signed error message is unicast by a node indicating the source and target nodes S and T:

$$A \rightarrow B : (ERR, S, T, cert_B, N_B, t, Sig_B)$$

V. Formal Model

It is believed that DoS attacks such as Black hole, Hello flood and Wormhole attacks are still possible in ARAN protocol and can lead to a significant loss of data. Therefore, the aim of this work is to apply formal model on ARAN using UPPAAL to detect its vulnerability to DoS attacks. While modelling ARAN it is assumed that the channel is ideal i.e. no message is lost because of collision or noise; the nodes are placed in rectangular grid and have the same radio range; and the density of network is limited to 4 maximum neighbouring nodes. In formal modelling one has to pay the price of high resources (computation, memory etc) so the assumptions are laid to simplify the model.

This section is further organized as follows: subsection V-A briefly describes the model of ARAN; subsection V-B describes the theorems or LTL properties applied on ARAN; subsection V-D and V-C describe detection of some DoS attacks detected by our formal frame work.

A. Proposed Model

The message format we employ in UPPAAL is as follows: Type, ID, TID, Source Certificate, Source Signature, Node Certificate, Node Signature. Type is the message type (i.e. rdp, rep or data); ID is used to address nodes in unicast messages (rdp or data) and remains null in broadcast(rdp); TID is the target ID to which the message is addressed (target in rdp and source in rep,data). The remaining fields are self explanatory. Note that we have not modelled the field nonce and time stamp since we have assumed that these will remain unchanged throughout due to public key signature and also since any attack will detect the change. These were removed to save state space. Therefore, the source signatures are based on TID and Source Certificate which serves the purpose of the encryption in model.

In UPPAAL, the node connectivity (RF links) is modelled using a NxN topology matrix with 1 or 0 in the matrix indicating existence or absence of a an RF link, where N is the total number of nodes in a network. A broadcast message passing in UPPAAL conventionally models WSN message passing. The 'channel' is thus modelled using a global flag to indicate whether a channel is busy or free. This flag can be used to check if a node is allowed to broadcast the message or not. Multiple receptions are modelled by another variable, *BusyNodes*. A node cannot transmit until this variable becomes 0 signalling that all recipients have performed their necessary actions and are now free. In UPPAAL, instead of using a quantified time (clocks), we use an event generator that sends a timeout to indicate that a phase has been completed. This is done to save state space. In general the complete UPPAAL model is split into 5 parts. These are the Event Generator model; Attacker model; Target model; Source model and Node model.

The ***Event Generator model*** has a task to generate different events in the protocol The event generator starts from the START location and generates an event sense data from the environment (SENSE_DATA). The nodes are informed of these events through the message and it models as a trigger that enables a node to do a specific job. Upon receiving this message the source node generates an RDP. The function MakeBlackholeAttacker() creates a black hole attack in system. If all the nodes become free along with the channel the event generator moves to FINSIH state.

The ***Source model*** starts in INITIATE_RDP phase and builds the RDP message when the Event Generator triggers an event message. It then broadcast RDP (SEND_RDP) to its neighbours. Note that the certificate field is replaced by the node ID in our model. The signature is then calculated using a global function MakeSignature() which simply adds the variables passed to it. Note also that a minor modification is done here to save state space i.e. the source node uses Cert and Sig fields as well which are NULL in ARAN. It has been done to make the messages symmetric and it does not cause any problem; it is just signing the signature again. The source node then moves to LISTEN phase and waits for the reply 'rep'. Upon receiving the reply (REC_REP) it checks if the signature is similar for both the sender and the target nodes (CHECK_SIGNATURE). If any of the 2 signatures are incorrect the model moves back to the LISTEN phase. If both signatures are correct then the source node unicast (SEND_DATA) the data message. A variable PathID has been used to save the ID of the first node which had authentically sent the reply message. The data is then unicast to that node. The source node again moves back to LISTEN phase.

The ***Target model***starts in LISTEN state. Upon receiving an RDP message, it checks three conditions: (i) whether the message is addressed to it (ii) the signature of sender is correct and (iii) whether the signature of the source node is correct (CHECK_SIGNATURE). If all three conditions are met, the target node replies in the reverse path (stored ID in SavedID variable) by sending REP message (SEND_REP). The target adds its own signature similarly as the source node had done initially while sending an RDP message. It then moves back to LISTEN phase. Upon receiving data message, the target again performs two tests i.e. checking the sender node's signature and the source node's signature (DATA_SIGNATURE). If both signatures are correct, the target model moves to SUCCESS location, otherwise it moves back to LISTEN phase.

The ***Node model***models all the intermediate nodes that are neither the sources nor the targets. The node model starts in LISTEN state and upon receiving any message (RDP, REP or DATA) it checks if the sender node's signature is correct. If the signature is correct, then the node rebroadcast that message by adding its own signature. It is worth noting that the signature and certificate of source/target pair as well as the Target ID remains unchanged in all cases. A variable SavedID stores the first node which has sent to the RDP so that it can send a REP later. Another variable PathID stores the first node which sends back the REP so that DATA can be sent to it later on.

The ***Attacker model*** comprises different attackers. For black hole attack the model is a simple node that does not forward data whereas the model is different for other attacks. As an example, the INA model simply duplicates the message it receives either RDP (DUPLICATE_RDP) or REP(DUPLICATE_REP) without adding itself. The attacker does not forward any data. A wormhole attacker is modelled by using a separate message tunnel that models the tunnel which lies between 2 wormhole nodes.

B. Verification

Once the model was built, simple theorems (LTL properties) were applied to confirm the protocol achieved the desired results. The theorems we applied performed sanity checks; and confirmed that safety and liveliness properties did hold in the protocol. The liveliness properties are defined as 'something good will eventually happen' while the safety properties are defined as 'nothing bad will ever happen' [2]. Thus data that never reaches the target can be termed as safety violation while the protocol deadlock is termed as liveliness failure. It is worth noting that in UPPAAL *E* and *A* indicate 'eventually' and 'always', respectively, while <> and [] are symbols for 'one case' and 'all cases', respectively. In addition, the symbols $\rightsquigarrow$ and $\Rightarrow$ indicate 'leads to' and 'implies' in UPPAAL theorems, respectively. More details about the LTL properties used in UPPAAL to verify the different theorems are available in [2]. The theorems/properties checked for ARAN protocol were:

Theorem 1 "RDP message is treated fairly:"
This is a sanity check to confirm that all nodes function in a correct manner at the route discovery phase. This theorem can be proved by the following LTL property:

$$Node_N.REC\_RDP \rightsquigarrow Node_N.SEND\_RDP$$

This property states that, when receiving the RDP message, all nodes N in the network will always rebroadcast the RDP infinitely. This means that if the RDP is received an infinite times it is sent an infinite times as well. Note that this process is only applied for node models and exclude target and source.

Theorem 2 "RREP message is treated fairly:"
This theorem is a sanity check and aims to confirm that all nodes operate correctly at the route reply phase. The theorem is proved by applying the following LTL property:

$$Node_N.REC\_REP \rightsquigarrow Node_N.SEND\_REP$$

The property states that all nodes N in network always rebroadcast the REP when receiving the REP message. Note that the target and source are immune to this task as they are not present in the property.

Theorem 3 "Legitimate node forwards data fairly:"
This theorem is a sanity check to confirm that all legitimate nodes forward data fairly; i.e. these do not drop data unless the node is a malicious black hole. The theorem is proved by checking the following LTL property:

$$(!Blackhole[N] \&\& Node_N.REC\_DATA) \rightsquigarrow Node1.SEND\_DATA$$

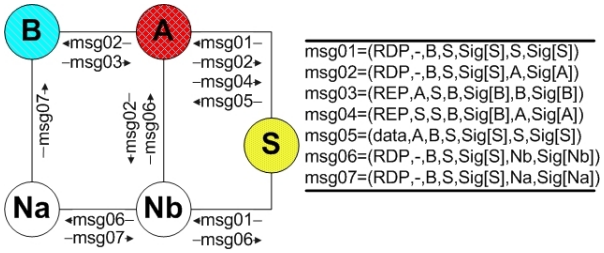

Fig. 2. Trace of UPPAAL showing that INA is possible in ARAN

The property states that all nodes N in the network will always rebroadcast the data unless it is a black hole node when the data message is received.

Theorem 4 "No deadlock in the ARAN protocol:"
This theorem is a liveliness check that something good will eventually happen; i.e. the protocol will always finish after it has started and never deadlocks in between. This theorem can be proved using following property:

$$EventGen.START \Rightarrow EventGen.FINISH$$

Note that EventGen can only reach the FINISH state if all nodes including the target and source become idle; i.e. they stop transmitting anything after the source has initiated the RDP. A more potent proof can be performed by applying the following property:

$$A[]notdeadlock$$

This property verifies that all the models never deadlock. However, our model does have a deadlock; so this property fails and the trace generated confirms that the model deadlocks with the Event Generator model in the FINISH state. We intentionally had introduced the deadlock to save on the state space. To confirm that the deadlock occurs because of this deliberate introduction and not due to any other reason, a self loop is placed in EventGen.FINISH location. Once done, the property did hold confirm that the model never deadlocks once it has reached the FINISH location which was our intended final location of the system.

Theorem 5 "Data from source node always reaches the target node:"
This final and most important theorem is the safety property which claims that when the system finishes, the target node will have successfully received data. The following LTL property is used to prove this theorem:

$$EventGen.FINISH \Rightarrow Target.SUCCESS$$

By applying different attack models to the ARAN model we have been able to confirm whether the theorem that checks the safety property is still valid in their presence. In the case of a safety property failure, ARAN is considered susceptible to that particular attack. Note that UPPAAL automatically generates a trace when any property fails. This trace informs us how and why an attacker has succeeded. In the next few sections we will discuss these attacks and the extent of their success with the trace showing how the attacks have succeeded.

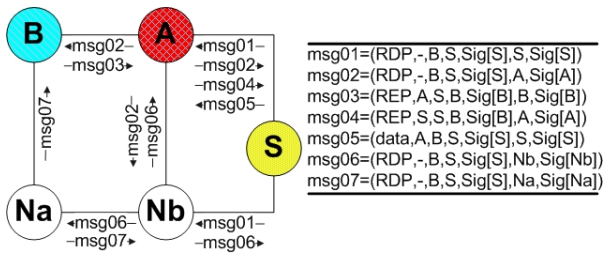

Fig. 3. Trace of UPPAAL showing that Black hole is possible in ARAN

C. Worm Hole Attack/INA

Figure2 shows a trace generated by UPPAAL in which the safety property fails in the presence of INA. Here node A is INA and retransmits messages msg01, msg02 and msg04 which it eavesdrops from its neighbours without adding itself. Note that node A does not duplicate data message (msg03). Thus, it creates a virtual link between nodes B and S. Node S then transmits data to node B which was not forwarded thus causing the safety property to fail. A similar behaviour was observed when the *wormhole* attacker was employed. However the legitimate RDP message (msg05) was rejected by the target node B as it had arrived late. This has also enabled *rushing attack*.

D. Black Hole Attack

Figure3 shows a trace generated by UPPAAL in which the theorem performing the safety property fails in the presence of a black hole attack. The black hole node A behaves normally when forwarding the RDP (msg02) and REP (msg03). But when node S transmits the data (msg04), the black hole node does not forward it. Note that the RDP message (msg07) from the node Na is ignored by the node B as it accepts the first arriving message i.e. via the black hole attacker.

VI. Simulation Results

Computer simulations only cannot guaranty that the protocol is immune from attacks even if the results show that. The reason is that the worst cases and hidden errors might be missed by a simulation. Thus, we have adopted a combination of formal modelling and computer simulations. The results obatained from formal modelling were applied to computer simulator TOSSIM [7]. TOSSIM operates at the bit level (high fidelity). Moreover TOSSIM simulation code can directly be programmed to hardware without any modification. We have implemented ARAN routing protocol. The encryption was simplified and public key details were removed as emphasis was on DoS attacks rather than encryption. It was, however, ensured that the message encrypted using a public key cannot be decrypted by the attacker. Our simulation results confirmed that DoS attacks such as black hole, INA and wormhole were successful in ARAN.

VII. Conclusions

ARAN is a robust routing protocol which employs public key cryptography. ARAN also does not consider the hop count rather it uses the fastest links to avoid attacks. Our formal framework, however, has confirmed that there are some weaknesses in ARAN where the cryptographic techniques cannot defeat INA and wormhole and by creating virtual links the routes can be corrupted. Our formal model has automatically detected this flaw in ARAN and generates a trace indicating why and how ARAN has failed in presence of INA/wormhole. Moreover, the nodes can be captured in real world applications and thus a compromised node might act as black hole attacker. This behaves normally in the normal routing operations and drops data packets. Our formal model also confirms the susceptibility of ARAN against the black hole attack. The model confirms that, in spite of data-signature mechanism, data might not reach the target nodes. We have shown that by using formal modelling hidden bugs in any routing protocol can be detected automatically and saving a significant time. Our future work involves testing more secure and robust routing protocols using formal modelling.

References

[1] T.R. Andel and A.Yasinsac. Automated evaluation of secure route discovery in MANET protocols. In K. Havelund, R. Majumdar, and J. Palsberg, editors, *Proceedings of 15th International SPIN Workshop on Model Checking Software (SPIN 2008),Los Angeles, CA, USA*, volume 5156 of *Lecture Notes in Computer Science*, pages 26–41. Springer, August 2008.

[2] G. Behrmann, A. David, K.G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of SysTems (QEST) 2006*, IEEE Computer Society, pages 125–126, 2006.

[3] Y. Hanna, H. Rajan, and W. Zhang. Slede: a domain-specific verification framework for sensor network security protocol implementations. In *Proceedings of the first ACM conference on Wireless network security (WISEC '08), Alexandria, VA, USA*, pages 109–118, March/April 2008.

[4] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99) , Seattle, Washington, USA*, pages 174–185, August 1999.

[5] G.J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison Wesley, 2004.

[6] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2-3):293–315, September 2003.

[7] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *First ACM Conference on Embedded Networked Sensor Systems (Sensys03)*, pages 126–137, November 2003.

[8] K. Saghar, W. Henderson, and D. Kendel. Formal modelling and analysis of routing protocol security in wireless sensor networks. In *PGNET '09*, June 2009.

[9] K. Saghar, W. Henderson, D. Kendel, and A. Bouridane. Formal modelling of a robustwireless sensor network routing protocol. In *Accepted for publication at NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2010)*, June 2010.

[10] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E.M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP 02)*, pages 78–89, November 2002.

[11] L. Tobarra, D. Cazorla, and F. Cuartero. Formal analysis of sensor network encryption protocol (snep). In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS 2007), Piscataway, NJ, USA*, pages 767–772, Pisa (Italy), October 2007.

[12] L. Tobarra, D. Cazorla, F. Cuartero, G. Diaz, and E. Cambronero. Model checking wireless sensor network security protocols: Tinysec + leap. In *Proc. of the First IFIP International Conference on Wireless Sensor and Actor Networks (WSAN'07)*, pages 95–106. IFIP Main Series, Springer, September 2007.